CS4430: Distributed Database Systems

Task: System Design

Due: 23/04/2022

Participants:

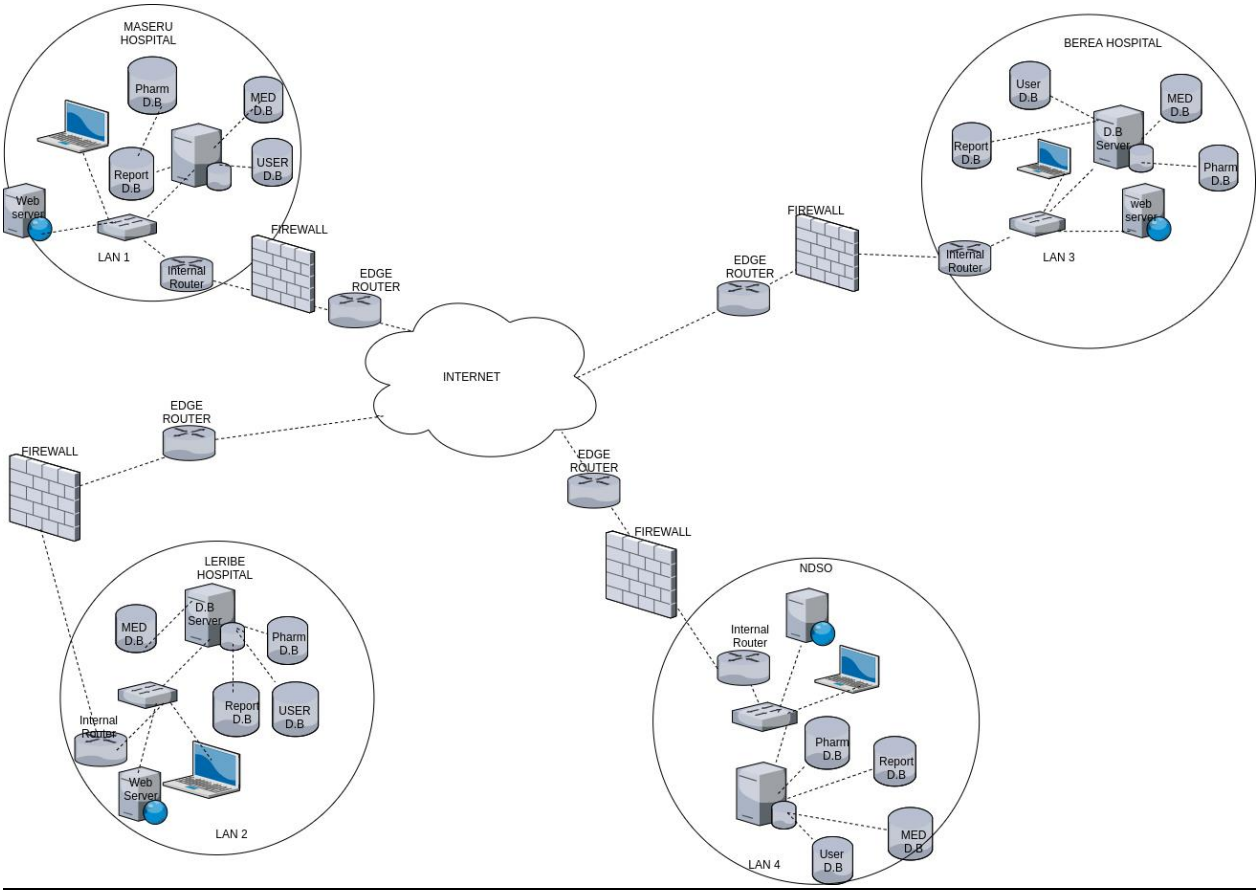
	Student Number	Surname, initials	Major
1.	201905468	Bale, K	B.Eng. systems and networks
2.	201602013	Rantsane, T.M	B.Eng. systems and networks
3.	201902318	Jockey, LM	B.Eng. systems and networks
4.	201902957	Seotsanyane, K.C	B.Eng. systems and networks

1. Detailed Real World Scenario System Architecture

Hardware Architecture

The hardware architecture consists of 4 laptops across all 4 sites with the following specs: Intel i7 processor, 16 GB of RAM, and 512 GB SSD storage. There should be a web server and database server in all sites.

Networking Architecture



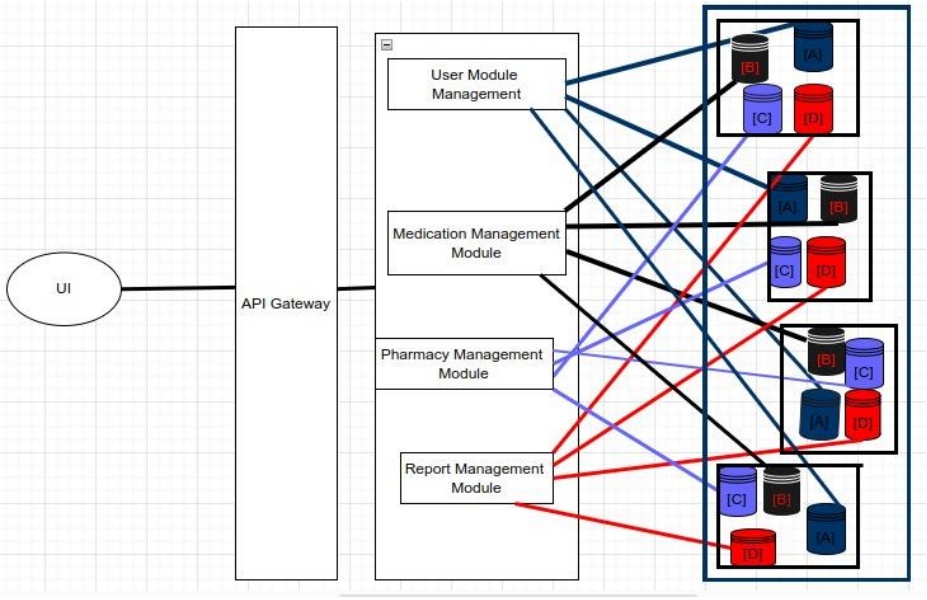
Software Architecture

The software architecture employs the use of *micro-service architecture*. There are 4 sites (Although could be more), being the Leribe Hospital, Berea Hospital, Maseru Hospital and NDSO sites. In each site, there are 4 databases, each handling a micro-service.

Micro-service A – User Management Module
Micro-service B – Medication Management Module
Micro-service C – Pharmacy Management Module
Micro-service D – Reporting Management Module

There is an API Gateway which acts as act as an intermediary between a client and the 4 micro-services, providing a single entry point for accessing various APIs.

User interface allows for different users to interact with system.



Rationale for design choice

The software architecture employs the use of *micro-service architecture.*

The reason for using this type of architecture:

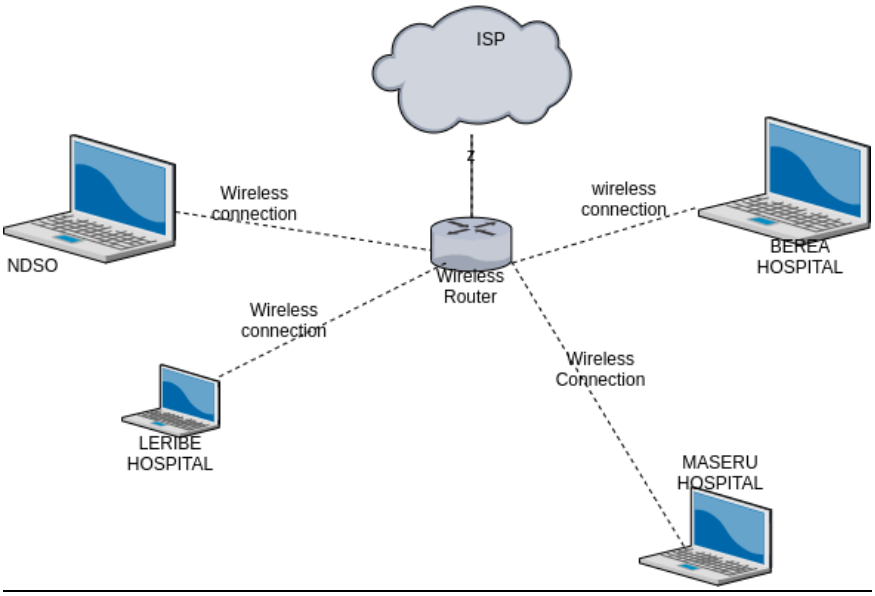
- Scalability: Micro-service architecture allows you to scale different parts of your application independently based on their resource needs. For example, if a particular micro-service is experiencing high traffic, you can scale it up without having to scale up the entire application.
- Agility: Micro-services are small, focused services that can be developed, deployed, and updated independently of each other. This makes it easier to introduce new features or update existing ones without affecting the entire application.
- Resilience: With microservices, if one service fails, it doesn't necessarily bring down the entire application. Other services can continue to function and provide value to users while the failing service is fixed.

2. Detailed Demonstration System Architecture

Hardware

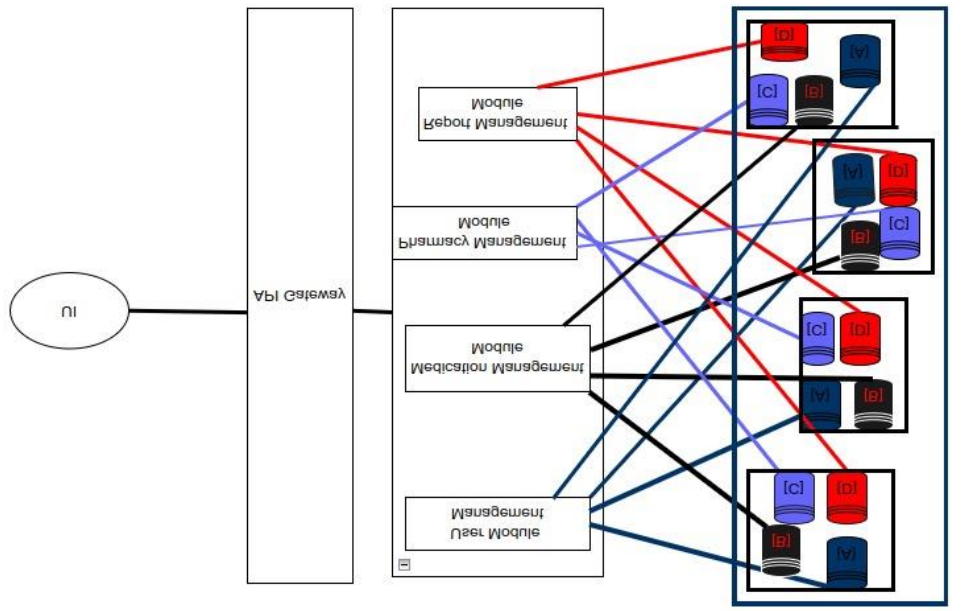
The hardware architecture consists of 3 local machines. The one machine is a laptop with an Intel i5 processor, 4GB of DDR4 RAM, and 256 GB SSD storage. The other is a laptop with an Intel Celeron processor, 4 GB of DDR3 RAM, and 500 GB HDD storage. The last one is a laptop with an Intel i5 processor, 4GB of DDR3 RAM, and 500GB HDD storage.

Network Architecture



Software Architecture

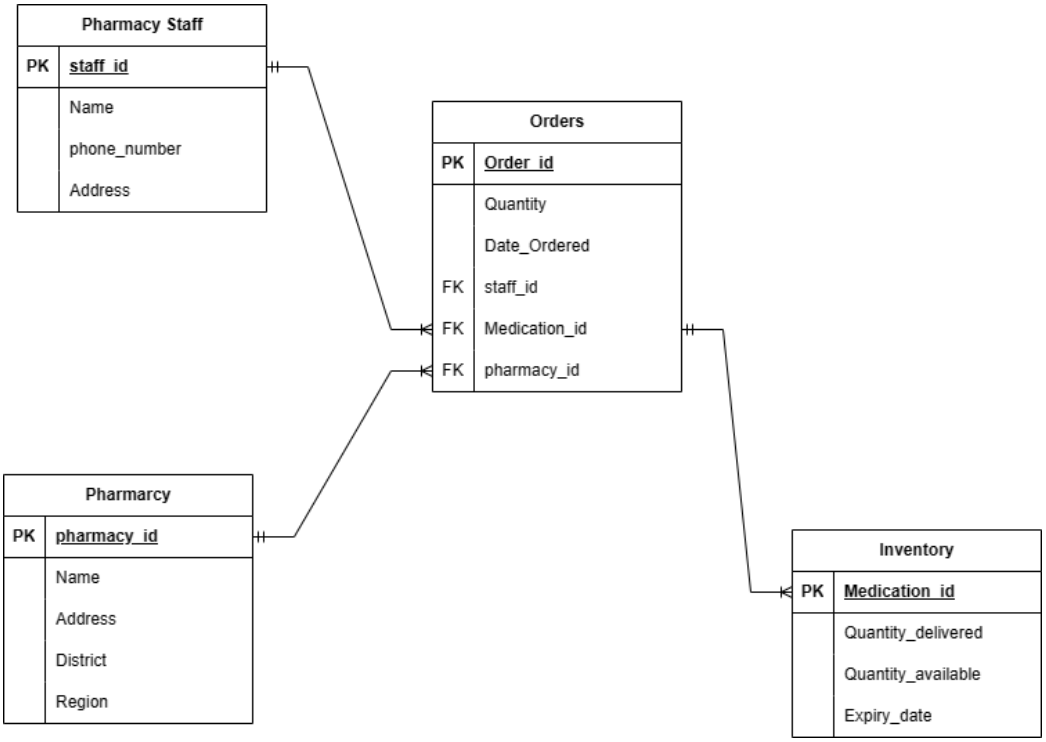
Same architecture as the real world scenario, difference is, all nodes are on a single machine.



3. Detailed Distributed Database Design

Pharmacy Management Module

GCS - Entity Relationship diagram:



Fragmentation

1. Orders table

Fragmenting Orders table using Derived Horizontal Fragmentation

The source relation: Location table fragmented using the location predicate, that is:

Simple predicate: p_1= pharmacy_id = "001"

After COM_MIN algorithm has been applied, the minterm predicates are:

m_1: pharmacy_id = "001"

m_2: pharmacy_id = "002"

m_3: pharmacy_id = "003"

m_4: pharmacy_id = "004"

Employing DHF upon Orders table using the predicate defined on the source relation, the fragments on Orders table will be:

F_1 for m_1: pharmacy_id = "001"

Order_id	Medication_id	Quantity(g)	Date_Ordered	Pharmacy_id	Staff_id
1	101	500	04/04/2023	001	1001
5	153	600	10/04/2023	001	1003
6	205	800	10/04/2023	001	1001

F_2 for m_2: pharmacy_id = "002"

Order_id	Medication_id	Quantity(g)	Date_Ordered	Pharmacy_id	Staff_id
3	256	2100	04/04/2023	002	1002
7	78	1000	14/04/2023	002	1002
9	324	1200	17/04/2023	002	1003

F_3 for pharmarcy_id = "003"

Order_id	Medication_id	Quantity(g)	Date_Ordered	Pharmacy_id	Staff_id
2	32	2100	03/04/2023	003	1001
11	14	1000	18/04/2023	003	1002
14	200	500	24/04/2023	003	1002

F_4 for pharmarcy_id = "004"

Order_id	Medication_id	Quantity(g)	Date_Ordered	Pharmacy_id	Staff_id
4	222	900	23/04/2023	004	1003
10	150	2000	23/04/2023	004	1002
13	340	1400	27/04/2023	004	1001

2. Inventory table

Fragmenting the table using Primary Horizontal Fragmentation:

Simple predicate

P_1 : Quantity_available <= " 500"

After COM_MIN algorithm has been applied, the minterm predicates are:

M_1: Quantity_available < = 500

M_2: Quantity_available > 500

Fragmented Tables :

F_1 for M_1: Quantity_available < = 500

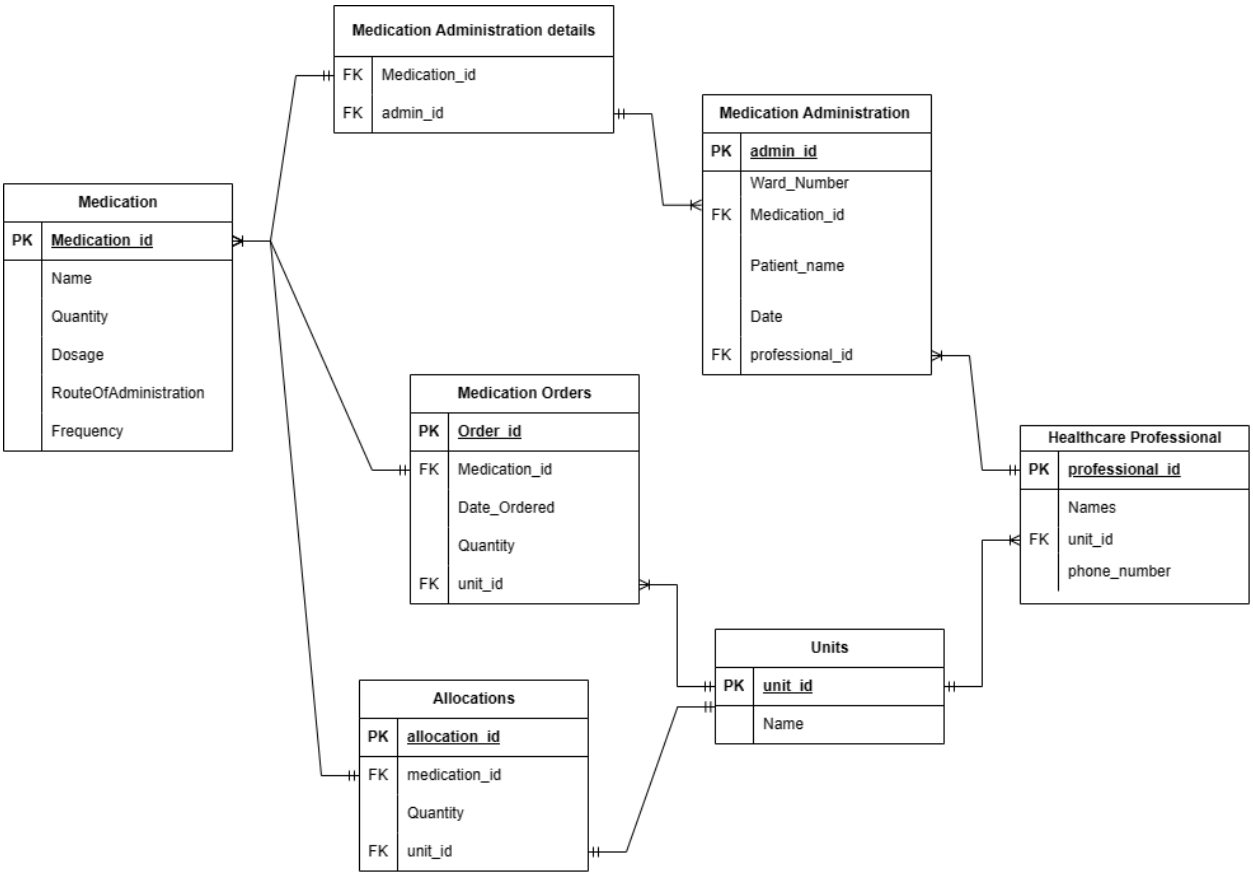
Medication_id	Name	Quantity_available(g)	Dosage	Expiry_date
101	Paracetamol	200	2 tablets, 3 times a day	31/12/2023
153	Flagyl	500	1 tablet, twice a day	30/07/2023
205	Metformin	357	2 tablets, twice a day	31/01/2024

F_2 for M_2: Quantity_available > 500

Medication_id	Name	Quantity(g)	Dosage	Expiry_date
102	Doxycycline	1050	1 tablet, twice a day	28/02/2024
103	Brufen	1253	1 tablets, 3 times a day	30/08/2023
104	Sinucon	680	1 tablets, once a day	31/05/2024

Medication Management Module

GCS- Entity Relationship diagram:



Fragmentation

1. Medication Administration table

Fragmenting Medication Administration table using Derived Horizontal Fragmentation

The source relation: Healthcare Professional table fragmented using the professional_id predicate, that is:

Simple predicate: p_1= professional_id = "001"

After COM_MIN algorithm has been applied, the minterm predicates are:

m_1: professional_id = "111"

m_2: professional_id = "112"

m_3: professional_id = "113"

m_4: professional_id = "114"

Fragmented Tables :

F_1 for M_1: professional_id = "111"

admin_id	Ward number	Medication_id	Patient_name	Date	Time	Professional_id
01	1	153	Lirontsho Mafantiri	02/04/2023	08 :00	111
05	2	205	Keke Bale	02/04/2023	14 :00	111

F_2 for M_2: professional_id = "112"

admin_id	Ward number	Medication_id	Patient_name	Date	Time	Professional_id
02	2	102	Moshe Rantsane	02/04/2023	14 :00	112
04	3	67	Keke Bale	03/04/2023	14 :00	112

F_3 for M_3: professional_id = "113"

admin_id	Ward number	Medication_id	Patient_name	Date	Time	Professional_id
09	3	80	Karabo Mokoena	03/04/2023	18 :00	113
17	4	102	Khabiso Lerata	04/04/2023	14 :00	113

F_4 for M_4: professional_id = "114"

admin_id	Ward number	Medication_id	Patient_name	Date	Time	Professional_id
08	1	65	Molula Botente	02/04/2023	08 :00	114
14	3	77	Naleli Mokhele	03/04/2023	18 :00	114

2. Medication Orders table

Fragmenting Medication Orders table using Derived Horizontal Fragmentation
The source relation: Units table fragmented using the unit_id predicate, that is:
Simple predicate: p_1= unit_id = "P11"
After COM_MIN algorithm has been applied, the minterm predicates are:
m_1: unit_id = "U11"
m_2: unit_id = "U22"
m_3: unit_id = "U33"
Resulting Fragments in the Medication Orders Table
F_1 for M_1: unit_id = “U11”
Medication Orders1

Order_id	Medication_id	Quantity(g)	Date_Ordered	Unit_id
1	108	50	07/04/2023	U11
2	105	60	04/04/2023	U11

F_1 for M_1: unit_id = “U22”
Medication Orders2

Order_id	Medication_id	Quantity(g)	Date_Ordered	Unit_id
3	244	100	03/04/2023	U22
4	333	200	07/04/2023	U22

Medication Orders3

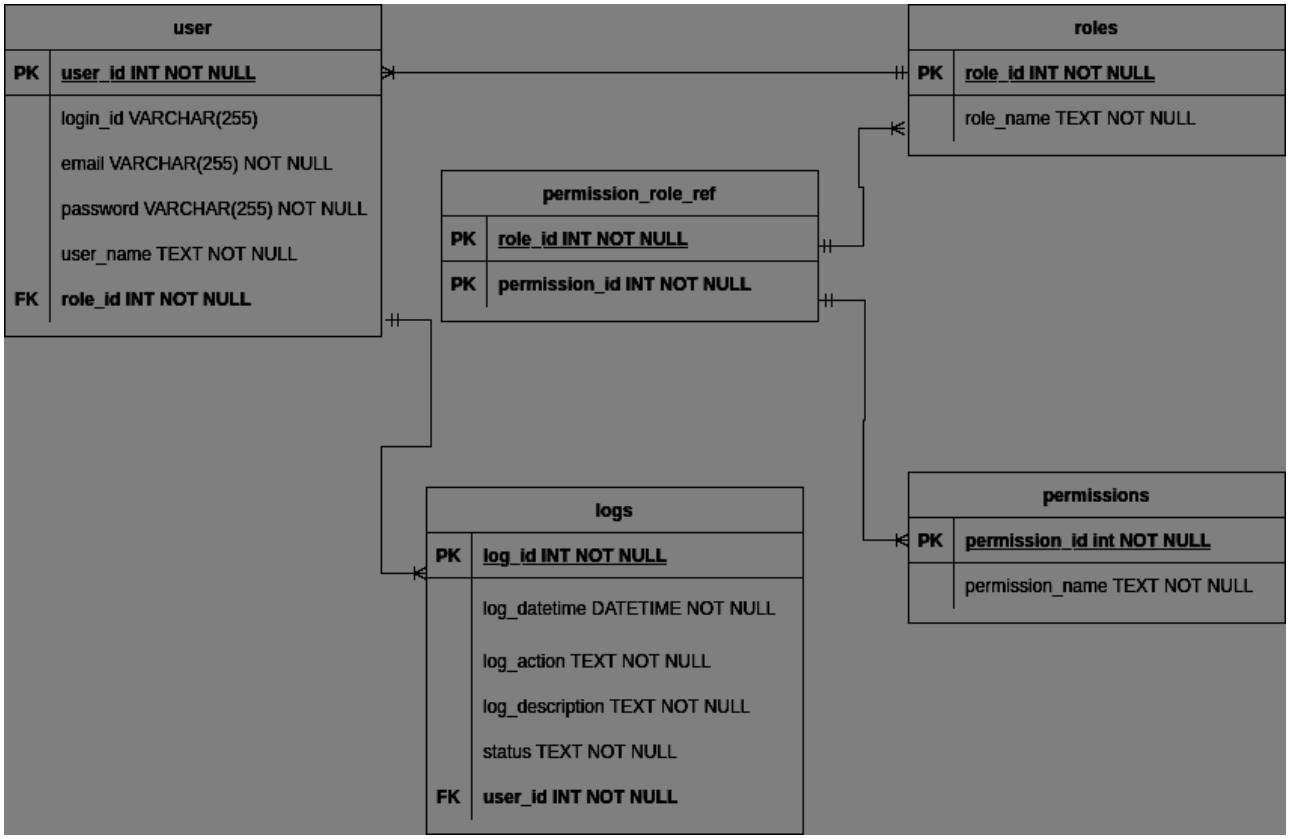
Order_id	Medication_id	Quantity(g)	Date_Ordered	Unit_id
5	222	255	02/04/2023	U33
6	134	300	06/04/2023	U33

These Medication Order fragments helps us store orders relating to one unit in separate table

USER MANAGEMENT MODULE

Pharmacy Management Module

GCS - Entity Relationship diagram:



Data Partitioning

1.User table

Fragmenting user table using Derived Horizontal Fragmentation:

Consider role_id as an edge joining users and roles table, note that target (role_id)=USER and source(role_id) = ROLES. Therefore, DHF is suitable for USER as follows:The fragments, **USER(A) = USER ⋉ ROLES(A), 1 ≤ A ≤ 4** ,Where, **ROLES(A)= σF(A)(ROLES), 1 ≤ A ≤ 2 where F = {role_id = 'Value'}**.Therefore,

USER = {USER ⋉ ROLES(1), USER ⋉ ROLES(2),USER ⋉ ROLES(3),USER ⋉ ROLES(4)} = {USER(1), USER(2),USER(3),USER(4)}

Demonstration using dummy data of the fragments:

USER(1)

user_id	login_id	email	password	user_name	**role_id**
1234	0001	tb@gmail.com	qqqwwww	Thibos	**101**
0987	0002	my@gmail.com	fbneokrbn	Monene	**101**

USER(2)

user_id	login_id	email	password	user_name	**role_id**
4587	0001	tb@gmail.com	qqqwwww	Thibos	**102**
2373	0002	my@gmail.com	fbneokrbn	Monene	**102**

USER(3)

user_id	login_id	email	password	user_name	**role_id**
8945	0001	tb@gmail.com	qqqwwww	Thibos	**103**
0389	0002	my@gmail.com	fbneokrbn	Monene	**103**

USER(4)

user_id	login_id	email	password	user_name	**role_id**
8953	0001	tb@gmail.com	qqqwwww	Thibos	**104**
2938	0002	my@gmail.com	fbneokrbn	Monene	**104**

It is important to note that the fragmentation of user table using role_id can help us to manage the access level for our users under user management module.

Logs table

Also logs table can be fragmented using Derived Horizontal Fragmentation, since user_id is used to join user and logs table. Now, Target(user_id)= logs and Source(user_id)=user , therefore DHF is suitable for LOGS as follows:

The fragments, LOGS**(A) = LOGS ⋉ USER(A), 1 ≤ A ≤ 4** ,Where, USER**(A)= σF(A)(USER), 1 ≤ A ≤ 2 , where F = {user_id = 'Value'}**.Therefore, LOGS = **{LOGS ⋉ USER(1), LOGS ⋉ USER(2), LOGS ⋉ USER(3),LOGS ⋉ USER(4)} = {LOGS(1), LOGS(2),LOGS(3),LOGS(4)}**

Demonstration using dummy data of the fragments:

LOGS(1)

Log_id	log_datetime	log_action	log_description	status	user_id
9348	2023-04-22 13:45:20	User login attempt	Valid details	success	1234
2350	2023-04-22 13:45:20	User login attempt	Wrong password	Access denied	1234

LOGS(2)

Log_id	log_datetime	log_action	log_description	status	user_id
5575	2023-04-22	User login	Valid details	success	5645

	13:45:20	attempt			
6788	2023-04-22 13:45:20	User login attempt	Wrong password	Access denied	5645

LOGS(3)

Log_id	log_datetime	log_action	log_description	status	user_id
1123	2023-04-22 13:45:20	User login attempt	Valid details	success	6745
3465	2023-04-22 13:45:20	User login attempt	Wrong password	Access denied	6745

LOGS(4)

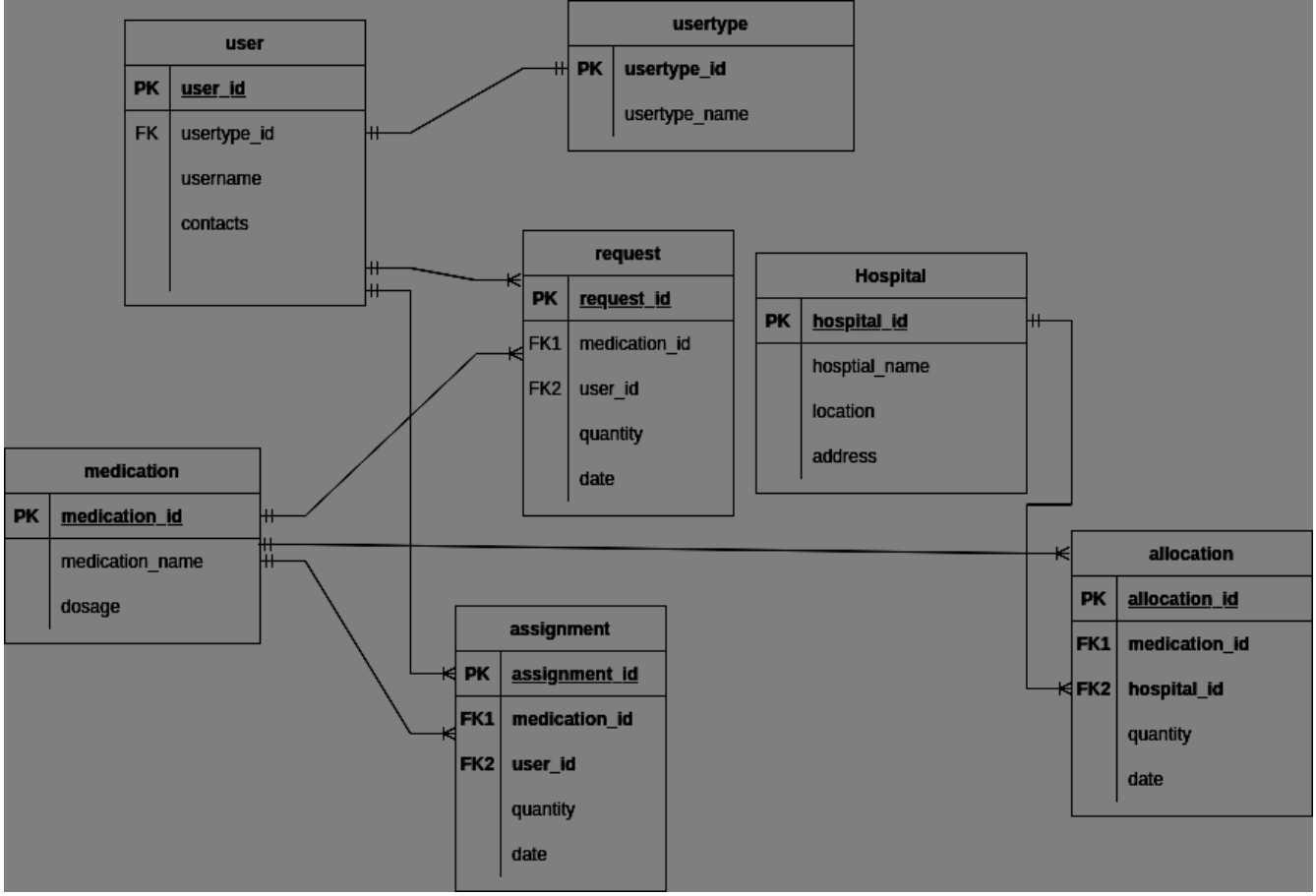
Log_id	log_datetime	log_action	log_description	status	user_id
3466	2023-04-22 13:45:20	User login attempt	Valid details	success	3667
2554	2023-04-22 13:45:20	User login attempt	Valid details	success	3667

The fragments of LOGS table can help to store log records for each user separately in different physical tables, based on the user_id value. This can help reduce contention and improve query performance for log records related to specific users.

REPORTING MANAGEMENT MODULE:

Pharmacy Management Module

GCS - Entity Relationship diagram:



-Data partitioning:

1. Allocation table

This table can be fragmented using Derived Horizontal Fragmentaion. Target (hospital_id)=ALLOCATION and source(hospital_id) = HOSPITAL. Therefore, DHF is suitable for ALLOCATION as follows:The fragments, ALLOCATION**(A) = ALLOCATION ⋉ HOSPTAL(A), 1 ≤ A ≤ 4** ,Where, HOSPITAL**(A)= σF(A)(HOSPITAL), 1 ≤ A ≤ 2 where F = {hospital_id = 'Value'}**.Therefore,

ALLLOCATION = {ALLOCATION ⋉ HOSPITAL(1), ALLOCAITON ⋉ HOSSPITAL(2),ALLOCATION ⋉ HOSPITAL(3),ALLOCATION ⋉ HOSPITAL(4)} = {ALLOCATION(1), ALLOCATION(2),ALLOCATION(3),ALLOCATION(4)}

Demonstration using dummy data of the fragments:

ALLOCATION(1)

Allocation_id	Medication_id	**hospital_id**	quantity	date
4579	3829	**4753**	653	23-04-2023
3748	3743	**4753**	237	10-04-2023

ALLOCATION(2)

Allocation_id	Medication_id	**hospital_id**	quantity	date
6543	2389	**3453**	653	23-04-2023
8765	9823	**3453**	237	10-04-2023

ALLOCATION(3)

Allocation_id	Medication_id	**hospital_id**	quantity	date

6543	2389	**3453**	653	23-04-2023
8765	9823	**3453**	237	10-04-2023

ALLOCATION(4)

Allocation_id	Medication_id	**hospital_id**	quantity	date
6543	2389	**6645**	653	23-04-2023
8765	9823	**6645**	237	10-04-2023

The fragments can help NDSO keep track of medication distributed among hospitals and we can make reports based on such information.

2. Request table.

This table can also be fragmented using Derived Horizontal Fragmentation. Target (user_id)=REQUEST and source(user_id)=USER.Therefore, DHF is suitable for REQUEST as follows:The fragments, REQUEST**(A) = REQUEST ⋉ USER(A), 1 ≤ A ≤ 4** ,Where, USER**(A)= σF(A)(USER), 1 ≤ A ≤ 2 where F = {user_d = 'Value'}**.Therefore,

REQUEST = {REQUEST ⋉ USER(1), REQUEST ⋉ USER(2),REQUEST ⋉ USER(3),REQUEST ⋉ USER(4)} = {REQUEST(1), REQUEST(2),REQUEST(3),REQUEST(4)}

Demonstration using dummy data of the fragments:

REQUEST(1)

request_id	**user_id**	medication_id	quantity	date
5748	**2343**	5483	543	24-05-2023
8433	**2343**	8754	655	04-06-2023

REQUEST(1)

request_id	**user_id**	medication_id	quantity	date
5748	**9876**	5483	543	24-05-2023
8433	**9876**	8754	655	04-06-2023

REQUEST(3)

request_id	**user_id**	medication_id	quantity	date
5748	**3456**	5483	543	24-05-2023
8433	**3456**	8754	655	04-06-2023

REQUEST(4)

request_id	**user_id**	medication_id	quantity	date
5748	**7654**	5483	543	24-05-2023
8433	**7654**	8754	655	04-06-2023

Allocation

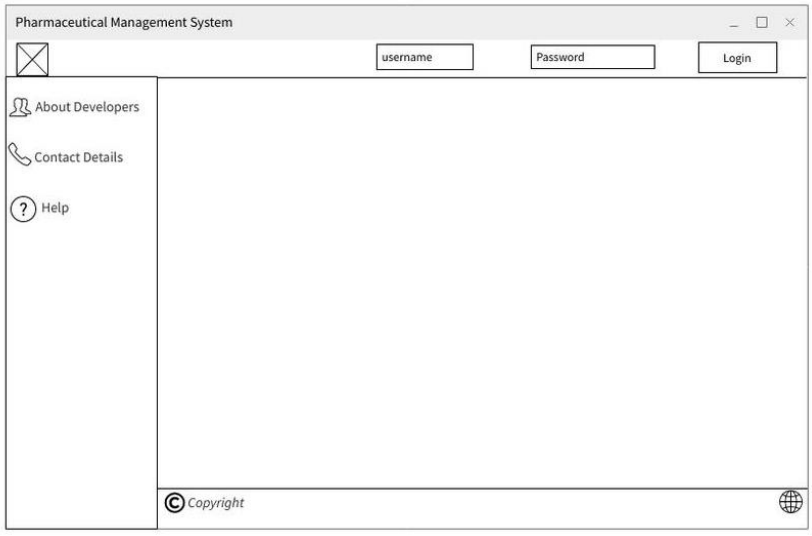
Since there are 4 sites, all fragments will be distributed evenly across all nodes using a Hashing Algorithm.

Replication

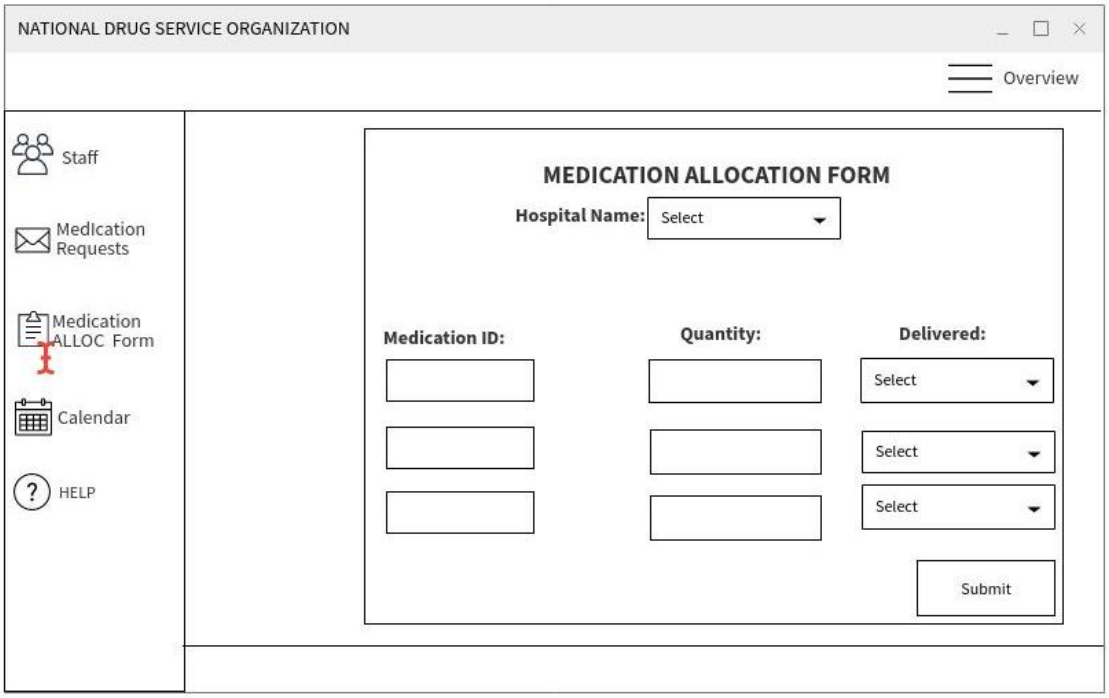
Replication factor = 2 across

UI/UX

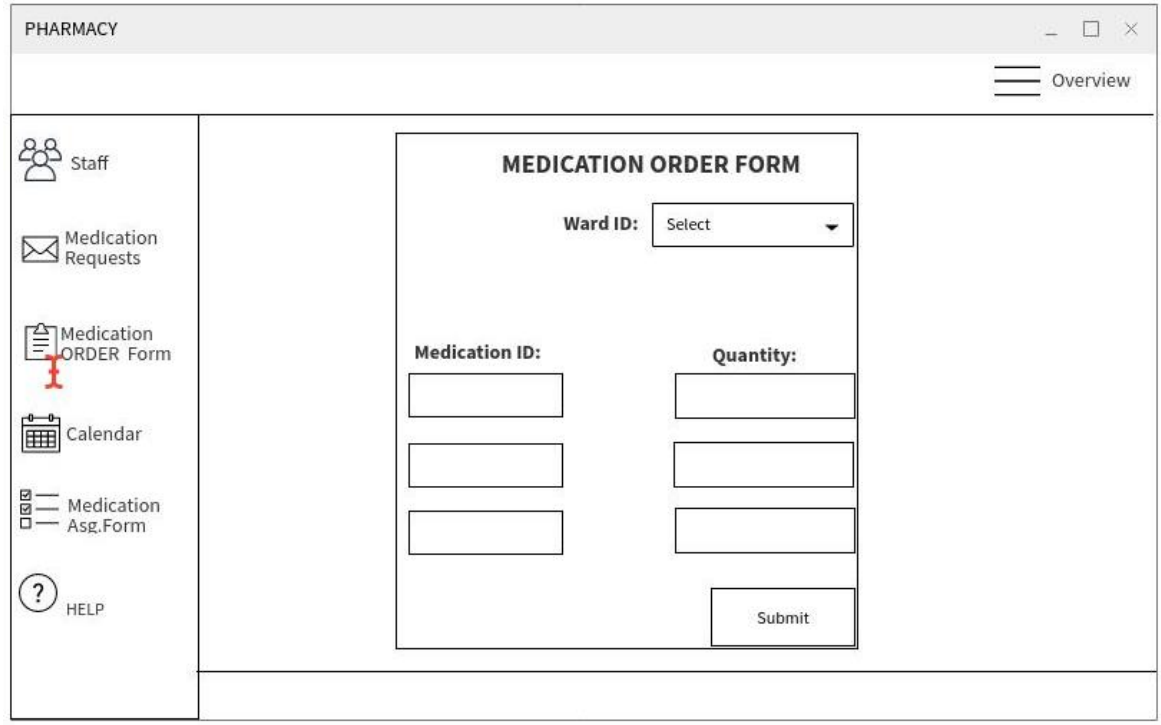
The login page. Different personnel will have different access rights, and view.



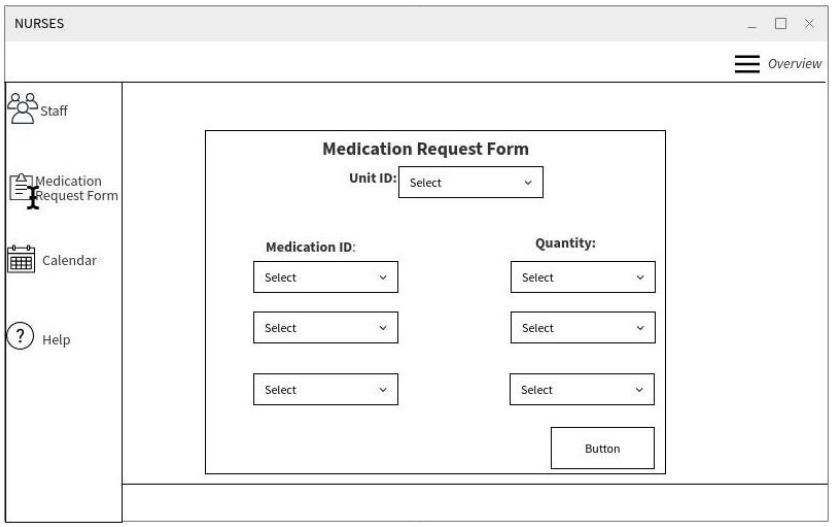
NDSO Personnel view- Allocation form to different hospital



Pharmacist View- Order form to order medication from NDSO



Nurse View- Medication Request from unit



TEST PLAN

Functionality	Check	Comment
1. System allows NDSO personnel to allocate medication to hospitals/pharmacies		
2. System allows pharmacies to confirm allocations from NDSO		
3. Allows pharmacists to make orders to NDSO		

4. Allows nursing unit to make orders from the pharmacy		
5. Allows nursing unit to confirm its medication allocation		
6. Allows nurses to document their administration of medication to patients		
7. System gives a warning when inventory levels are low		
8. System is user-friendly		
9. System is able to handle high volumes of data		